REMARKS

The present response amends claims 1 and 6-12. Claims 1-15 remain pending in the captioned case. Further examination and reconsideration of the presently claimed application are respectfully requested.

Section 112 Rejection

Claims 1-15 were rejected under 35 U.S.C. § 112, first paragraph, as allegedly failing to comply with the written description requirement. Applicant respectfully disagrees. The proper methodology for determining adequacy of written description is set forth in MPEP 2163. Specifically, "the Examiner has the initial burden . . . of presenting evidence or reasons why a person skilled in the art would not recognize that the written description of the invention provides support for the claims." MPEP 2163(II)(A). There is a strong presumption that an adequate written description of the claimed invention is present in the specification as filed. *In re Wertheim*, 541 F.2d. 257 (CCPA 1976). Only if claims are added or amended must Applicant then show support in the original disclosure in order to fulfill the requirements of 35 U.S.C. § 112. MPEP 2163(II)(A).

First, the rejected claims were a part of the original specification. Second, since the examined claims were in the original application, the strong presumption of an adequate written description remains unless an Examiner can rebut that "strong presumption." MPEP 2163. Applicant contends that the Examiner has not fulfilled the requirements of MPEP 2163. Specifically, the unsupported allegation on page 2 of the Office Action that alleges "the present technology does not support running multiple operating environments in the same machine" is in no way relevant to the actual terminology claimed. A closer examination of independent claims 1, 6, and 13 reveals code portions (first and second code portions) that can be run on different operating environments. Nowhere in the present claims is there any reference to running multiple operating environments "in the same machine." However, if the Examiner believes this is how the claim should be construed, then certainly the same machine can operate a first operating system and later, a second operating system. If so, then the machine is altered by the operating systems it runs -- both of which can execute the claimed first and second portions.

It would be inappropriate to construe claims 1, 6, and 13 with terminology not present in those claims. More specifically, the Examiner appears to construe the claims as somehow involving a "computer network." The Office Action, page 2, states: ". . . the system must be looked upon as a computer network in order to support more than one operating environment." To offer such a construction would be inappropriate not only from the perspective of the claims as part of the original disclosure, but the disclosure itself. To offer this inappropriate form of claim interpretation violates the very purpose of 35 U.S.C. § 112, which mandates the Examiner analyze each claim and "determine what the claim as a whole covers." MPEP 2163(II)(A)(1). Independent claims 1, 6, and 13 in no way mention a computer network, a client, a server, or a transmission path therebetween. Instead, these claims merely indicate code portions that illicit action from two different operating environments. To construe otherwise not only runs counter to the claim meaning and description as originally presented, but also fails to meet the threshold burden of reading and understanding the specification and claims as written.

For at least these reasons, Applicant respectfully traverses this rejection and requests that it be removed in its entirety.

Section 102 Rejection

Claims 1, 2, 5-7, 12, 13, and 15 were rejected under 35 U.S.C. § 102(b) as being anticipated by "Java, RMI and CORBA," by David Curtis (hereinafter "Curtis"). The standard for "anticipation" is one of fairly strict identity. A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art of reference. *Verdegaal Bros. v. Union Oil Co. of California*, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987); MPEP 2131. Curtis does not disclose all limitations of the currently pending claims, some distinctive features of which are set forth in more detail below.

Curtis does not teach or suggest code that illicits action from two different operating environments/systems. Present claims 1 and 13 each recite a first code portion and a second code portion. The first code portion illicits action from a Java virtual machine. The second code portion illicits action from a runtime operating environment. In addition to the first and second code portions, a third code portion is derived from the second code portion. The third code portion illicits action from a second operating environment different from the runtime operating environment.

Instead of having a first code portion that illicits action from a JVM, a second code portion that illicits action from a runtime operating environment, and a third code portion that illicits action from a second operating environment, Curtis appears to describe on page 2 a code portion that illicits action only from a JVM (Curtis -- pg. 2, lines 1-9). Specifically, Curtis states that "Java has its own, built-in native ORB" (Curtis -- pg. 2, line 1), and that "Java makes it impractical for use with objects or applications written in other language." (Curtis -- pg. 2, lines 8-9). Since the code in Curtis is written in a native ORB language, only the operating environment which recognizes ORB will be activated. Any other operating environments activated from another code portion (other than Java language having its own native ORB) is nowhere taught or suggested in Curtis.

While Curtis describes on page 3, Fig. 2, a client-server architecture using RMI, Curtis specifically explains that "RMI is a Java-to-Java technology." (Curtis -- pg. 3, line 2.) While, for example, a remote object such as C++ can be adapted to Java language, only the adapted Java language is sent from or to the server -- not a combination of a Java language and a second object code that is not Java as in present independent claims 1 and 13.

To understand what is presently claimed, a tutorial of Java languages and the problems of Java would be beneficial. As set forth in the present specification, Java is a programming language that can create objects from what are known as "classes." (Specification -- pg. 2, line 12 - pg. 4, line 6.) One benefit of Java is that many different application programs can be converted or "translated" into instructions specific to a target machine, such instructions are oftentimes referred to as "native code." (Specification -- pg. 4, lines 18-22.) Unfortunately, however, the portability of Java does not guarantee that the graphical output of the GUI when executing those languages will have the same "look-and-feel." (Specification -- pg. 5, lines 5-22.)

To solve this problem, the present invention was contemplated. Specifically, a Java API oftentimes referred to as "AWT" has been derived that will not only bridge the gap between portable Java code and native code, but will also present the same object across different host platform interfaces (HPIs) with the same look-and-feel (Specification -- pg. 8, line 1 - pg. 10, line 16). One way in which to perform look-and-feel consistency is to segregate code that is dependent on the operating system from code which is not. The dependent code is purposely written as native code (e.g., C/C++), whereas the independent code can be maintained as Java code. A common HPI will receive the component peer classes written in Java and native code regardless of the operating environment currently being used. The native code is, therefore, the second

code portion and the third code portion. For example, the second code portion can be code indigenous to Win32 operating environment, and the third code portion can be indigenous to Solaris operating environment. Of course, numerous other operating environments can be used and are contemplated as the claimed "runtime operating environment" and "second operating environment."

Nowhere is there any mention in Curtis of two code portions that can be used to adapt to different operating environments. In fact, there is no mention whatsoever in Curtis of any operating environment, much less two different operating environments. Accordingly, Applicant respectfully requests removal of this rejection as it pertains to independent claims 1 and 13, as well as claims dependent therefrom.

Curtis does not teach or suggest a third code portion derived from a second code portion, or that the third code portion illicits action from a second operating environment different from a runtime operating environment on which the second code portion illicits action. Each of the present independent claims 1 and 13 not only define different code portions operating on different environments, but the third code portion must be derived from a second code portion, each of which operate on different environments. Not only does Curtis fail to disclose any code portion derived from another code portion, but certainly does not mention the non-derived and derived code portions operating on different environments. Accordingly, Applicant asserts that for this additional reason, this rejection must be removed in its entirety.

Curtis does not teach or suggest transforming a second code portion into a plurality of machine executable code modules, each of which is operable on only one of a plurality of operating systems. Present claim 6 defines not only first and second code portions, but the first code portion being indigenous to an operating system independent virtual execution function and the second code portion transformed into a plurality of machine executable code modules. Each of the machine executable modules is operable on only one unique operating system among a plurality of operating systems. Support for the amendments to claim 6 are found throughout the present specification, and illustrated in Figs. 2 and 3 and the description pertaining to same. Absent any suggestion for transforming the second code portion into machine executable modules, Curtis cannot be an anticipatory reference to independent claim 6 or claims dependent therefrom. Accordingly, Applicants respectfully request removal of this rejection.

Section 103 Rejection

Claims 3, 4, 8-11, and 14 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Curtis in view of "The AWT Native Interface," by Sun Microsystems, Inc. (hereinafter "Sun"). To establish a case of *prima facie* obviousness of a claimed invention, all of the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d. 981 (CCPA 1974); MPEP 2143.03. Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so. *In re Fine*, 837 F.2d. 1071 (Fed.Cir. 1988); MPEP 2143.01. The cited art does not teach or suggest each and every limitation of the independent claims and, moreover, the cited art cannot be combined as proposed in the Office Action. However, even if the hypothetical combination can be made, some distinctive features of the current claims are set forth in more detail below.

Curtis and Sun, either singularly or in combination, do not suggest first and second code portions written in Java and C/C++ programming languages, or that the C/C++ programming language is written from Java AWT Component Peer classes. Present dependent claims 3, 4, 8, and 11 each define the specific programming languages set forth in the first and second code portions. The combination of the first code portion (and not the second) written in Java language, and the second code portion (and not the first) written in C/C++ is neither taught nor suggested in either Curtis or Sun. In order to render the present claims obvious, there must be some passage within each of the cited references that would suggest to a skilled artisan that code can be demarcated into first and second portions, and that the first and second portions are written in different languages -- specifically Java and C/C++. There is no such teachings in either Curtis or Sun. Moreover, there is no teaching in either Curtis or Sun that the C/C++ programming language must be written from Java AWT Component Peer classes.

Absent any suggestion of demarcating code for different languages or writing Java AWT Component Peer classes in C/C++ language, Applicant must respectfully request that this rejection be removed in its entirety.

CONCLUSION

The present amendment and response is believed to be a complete response to all issues raised in the Office Action mailed October 3, 2003. The prior art made of record, but not relied upon, is not considered pertinent to the captioned application. In view of the remarks traversing the rejections, Applicant asserts that pending claims 1-15 are in condition for allowance. If the Examiner has any questions, comments or suggestions, the undersigned earnestly requests a telephone conference.

No fees are required for filing this amendment; however, the Commissioner is authorized to charge any additional fees which may be required, or credit any overpayment, to Conley Rose, P.C. Deposit Account No. 03-2769/5468-07900.

Respectfully submitted,

Mollie E. Lettang

Mollie E. Lettang
Reg. No. 48,405
Agent for Applicant(s)

Conley Rose, P.C.
P.O. Box 684908
Austin, TX 78768-4908
(512) 476-1400
Date: January 5, 2004
KLD